1、重绘和回流

在浏览器地址栏里面输入地址（URL）到整个页面加载完成的步骤？（全栈）

浏览器解析html文件流程：

1.解析html文件会创建一个DOM树

2.解析css，创建一个css规则树

3.将DOM树和css规则树结合，创建一个渲染树（render tree）

4.根据渲染树去绘制页面

重绘和回流（重排）

重绘：页面中的标签的简单的外观样式发生改变，不会影响页面的整体布局，就会引起重绘

回流（重排）：页面中的标签的样式（会让整个页面布局发生改变的样式）发生改变就会引起回流（重排）

回流一定会引起重绘，重绘不一定会引起回流

最小化重绘和回流：

1.针对样式：一个标签有多个样式需要修改，则可以使用修改类名的方法达到一次性修改

2.通过js页面中添加内容：采用虚拟DOM，一次性添加，不要去添加多次

3.动画：让这个要产生动画的标签脱离标准文档流

2、垃圾回收

1、GC（Garbage Collection）：

程序工作过程中会产生垃圾，也就是程序不用的内存或者是用过了，以后不会再用的内存空间，GC负责回收垃圾。

JS引擎发现、清理垃圾：

js的引用数据类型是保存在堆内存中的，然后在栈内存中保存一个对堆内存中实际对象的引用。

```
let demo = {
    name:"andy"
};
demo = [1,2,3,4];
```

声明一个变量demo，引用对象｛name:"andy"｝，把这个变量重新赋值一个数组对象，之前的对象引用关系就没有了。

2、垃圾回收策略

2.1 标记清除法

算法分为 标记 和 清除 两个阶段，标记阶段即为所有活动对象做上标记，清除阶段则把没有标记（也就是非活动对象）销毁。

过程：

- 垃圾收集器在运行时会给内存中的所有变量都加上一个标记，假设内存中所有对象都是垃圾，全标记为0
- 然后从各个根对象开始遍历，把不是垃圾的节点改成1
- 清理所有标记为0的垃圾，销毁并回收它们所占用的内存空间
- 最后，把所有内存中对象标记修改为0，等待下一轮垃圾回收

优点：

实现比较简单，打标记也无非打与不打两种情况，这使得一位二进制位（0和1）就可以为其标记，非常简单

缺点：在清除垃圾后，剩余的对象的内存位置是不变的，会导致空闲内存空间不连续，出现内存碎片；并且由于剩余空闲内存不是一整块，而是由大小不同的内存组成的内存列表，于是就牵扯出内存分配的问题。

内存碎片化，空闲内存块是不连续的，容易出现很多空闲内存块，还可能会出现分配所需内存过大的对象时找不到合适的块；

分配速度慢，因为即便是使用 First-fit 策略，其操作仍是一个 O(n) 的操作，最坏情况是每次都要遍历到最后，同时因为碎片化，大对象的分配效率会更慢。

3、引用计数

引用计数（Reference Counting），把 对象是否不再需要 简化定义为 对象有没有其他对象引用到它，如果没有引用指向该对象（零引用），对象将被垃圾回收机制回收。

过程：跟踪记录每个变量值被使用的次数

- 当声明了一个变量并且将一个引用类型赋值给该变量的时候这个值的引用次数就为 1
- 如果同一个值又被赋给另一个变量，那么引用数加 1
- 如果该变量的值被其他的值覆盖了，则引用次数减 1
- 当这个值的引用次数变为 0 的时候，说明没有变量在使用，这个值没法被访问了，回收空间，垃圾回收器会在运行的时候清理掉引用次数为 0 的值占用的内存

优点：

引用计数算法的优点，对比标记清除来看就会清晰很多，首先引用计数在引用值为 0 时，也就是在变成垃圾的那一刻就会被回收，所以它可以立即回收垃圾

缺点：

引用计数的缺点，首先它需要一个计数器，而此计数器需要占很大的位置，因为不知道被引用数量的上限，还有就是无法解决循环引用无法回收的问题，这也是最严重的问题

3、防抖和节流

防抖：针对高频触发事件，让其事件处理函数在一段时间后执行， 如果在这段时间内容再次触发，则重新开始计时, 直到在这段时间内没有再次触发，最后才去执行

节流：针对高频触发事件，让其事件处理函数每隔一段时间执行一次，稀释执行次数

防抖 eg：

```
    <button id="debounce1">点我防抖！</button>

    <script>
        window.onload = function () {
            // 1、获取这个按钮，并绑定事件
            var myDebounce = document.getElementById("debounce1");
            myDebounce.addEventListener("click", debounce(handle));
        }

        // 2、防抖功能函数，接受传参
        function debounce(fn) {
            // 4、创建一个标记用来存放定时器的返回值
            let timeout = null;
            return function () {
                // 5、每次当用户点击/输入的时候，把前一个定时器清除
                clearTimeout(timeout);
                // 6、然后创建一个新的 setTimeout，
                // 这样就能保证点击按钮后的 interval 间隔内
                // 如果用户还点击了的话，就不会执行 fn 函数
                timeout = setTimeout(() => {
                    fn.call(this, arguments);
                }, 1000);
            };
        }
        // 3、需要进行防抖的事件处理
        function handle() {
            // 有些需要防抖的工作，在这里执行
            console.log("防抖成功！");
        }

    </script>
eg2:
<div class="demo"></div>

    <script>
    // 获取节点
        let demo = document.querySelector('.demo');
    // 回调函数
        function myfn() {
```

```
            console.log('1111');
            console.log(this);
        }
    // 事件、调用函数
        demo.onmousemove = debonuce(myfn, 1000);
    // 防抖函数
        function debonuce(fn, delay) {
            let flag;
            return function () {
                if (flag) clearInterval(flag);
                flag = setTimeout(() => {
                    console.log(this);
                    fn();
                    // fn.call(this);
                }, delay)
            }
        }
    </script>
```

节流 eg:

```
    <button id="throttle">点我节流! </button>

    <script>
        window.onload = function () {
            // 1、获取按钮，绑定点击事件
            var myThrottle = document.querySelector("#throttle");
            myThrottle.addEventListener("click", throttle(sayThrottle));
        }

        // 2、节流函数体
        function throttle(fn) {
            // 4、通过闭包保存一个标记
            let canRun = true;
            return function () {
                // 5、在函数开头判断标志是否为 true，不为 true 则中断函数
                if (!canRun) {
                    return;
                }
```

```
                // 6、将 canRun 设置为 false，防止执行之前再被执行
                canRun = false;
                // 7、定时器
                setTimeout(() => {
                    fn.call(this, arguments);
                    // 8、执行完事件（比如调用完接口）之后，重新将这个标志设置为 true
                    canRun = true;
                }, 1000);
            };
        }

        // 3、需要节流的事件
        function sayThrottle() {
            console.log("节流成功! ");
        }

    </script>

    eg2:
       // 1 获取标签
        let demo = document.querySelector('.demo');

        // 2 绑定鼠标移动事件
        demo.onmousemove = throttle(myfn, 1000);

        // 3 需要节流的事件
        function myfn(el) {
            console.log('1111');
            console.log(this); // this: <div class="demo"></div>
        }

        // 4 封装节流函数
        function throttle(fn, delay) {
            // 5 通过闭包保存一个标记
            let flag = null;
            return function (ev) {  // 节流函数中真正的处理函数
                // 在函数开头判断标志是否为 null，不为 null 则中断函数
                if (flag) return false;

                let e = ev || event;
```

```
                let _this = this;
                flag = setTimeout(() => {
                    // fn();
                    console.log(this); // this: <div class="demo"></div>
                    fn.call(_this, e);
                    flag = null;
                }, delay)
            }
        }
    </script>
```

4、懒加载和预加载

```
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        .content {
            height: 1200px;
            background-color: #ccc;
        }

        .test {
            background-color: pink;
        }
    </style>

</head>

<body>
    <button>点击button加载图片</button>
    <div class="content">
        <img src="" alt="">
    </div>
    <div class="test">
        <img src="" alt="" class="img">
```

```
        <img src="" alt="" class="img">
        <img src="" alt="" class="img">
        <img src="" alt="" class="img">
        <img src="" alt="" class="img">
    </div>

    <script>
        // 预加载
        let img = new Image();
        let contentImg = document.querySelector('.content img');
        img.src = 'https://img2.baidu.com/it/u=1513943675,1427588238&fm=253&fmt=auto&app=120&f=JPEG?w=600&h=400';
        document.querySelector('button').onclick = function () {
            contentImg.src = 'https://img2.baidu.com/it/u=1513943675,1427588238&fm=253&fmt=auto&app=120&f=JPEG?w=600&h=400'
        }

        // 懒加载：页面滚动到test，就加载图片（可以提前一点加载）
        let test = document.querySelector('.test');
        let imgs = document.querySelectorAll('.img');
        window.onscroll = function () {
            // 打印test盒子距离窗口顶部的距离    窗口的高度
            console.log(test.getBoundingClientRect().top, window.innerHeight);

            if (test.getBoundingClientRect().top - 50 < window.innerHeight) { // -50是为了提前一点加载，不用滚动到时候在等待
                console.log('可以懒加载了');
                imgs.forEach(item => {
                    item.src = 'https://cdn.cnbj1.fds.api.mi-img.com/mi-mall/5713971c4bb6512743dfd06023080268.png?thumb=1&w=250&h=250&f=webp&q=90';
                })
            }
        }

    </script>
```

5、鼠标右键事件

```
<script>
    let box = document.querySelector('.box');
    let ul = document.querySelector('ul');
    box.oncontextmenu = function () {
        event.preventDefault();
        ul.style.display = "block";
        // 获取鼠标右键点击的位置
        console.log(event.clientX, event.clientY);
        ul.style.left = event.clientX + 'px';
        ul.style.top = event.clientY - this.getBoundingClientRect().y + 'px';
    }
    // 鼠标左键点击，隐藏ul菜单栏
    window.onclick = function () {
        ul.style.display = "none";
    }
</script>
```

6、鼠标拖拽事件

```
<script>
    let img = document.querySelector('img');
    img.ondragstart = function () {

    }
    window.ondragover = function () {
        event.preventDefault();
    }
    window.ondrop = function (e) {
        console.log(img.getBoundingClientRect());
        img.style.left = e.clientX + 'px';
        img.style.top = e.clientY + 'px';
    }

</script>
```

7、瀑布流

```
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        .demo {
            width: 1240px;
            margin: 0 auto;
            position: relative;
        }

        .test {
            background-color: pink;
            position: absolute;
            /* left: 0;
            top: 0; */
            margin-bottom: 10px;
            width: 240px;
        }
    </style>

</head>

<body>
    <div class="demo"> </div>

    <script>
        let data = [{ id: 0 }, { id: 1 }, { id: 2 }, { id: 3 }, { id: 4 }, { id: 5 }, { id: 6 }, { id: 7 }, { id: 8 }, { id: 9 }, { id: 10 }, { id: 11 }, { id: 12 }, { id: 13 }, { id: 14 }, { id: 15 }, { id: 16 }, { id: 17 }, { id: 18 }, { id: 19 }, { id: 20 }, { id: 21 }, { id: 22 }, { id: 23 }];
        let demo = document.querySelector('.demo');
        data.forEach((item, index) => {
            // 盒子高度：（随机）
            let testHeight = Math.ceil(Math.random() * (360 - 280) + 280);
            // 盒子左边距离：（每个盒子的宽度+间隙）*（同一排盒子的索引对5取余）
            let testLeft = (240 + 10) * (index % 5);
            // 盒子上边距：
```

```
            //（第一排为零，后面的索引大于4的盒子，也就是第二排以后的盒子，
            // 上边距减去第一排盒子的可视高度后+间隙+本身的y轴偏移量。
            let testTop = 0;
            if (index > 4) {
                let allTest = document.querySelectorAll('.test');
                testTop = allTest[index - 5].clientHeight + 10 + allTest[index - 5].offsetTop;
            }
            // 加载到页面中
            demo.innerHTML += `<div class="test" style="height:${testHeight}px;
                                left:${testLeft}px;top:${testTop}px;">${item.id}</div>`;
        })
    </script>
```

8、表单联动

```
    省：<select name="" id="province">
        <option value="">请输入省</option>
    </select>
    市：<select name="" id="city">
        <option value="">请输入市</option>
    </select>

    <script>
        let data = [{
            id: 101,
            province: "四川省",
            citys: [{
                id: 1011,
                city: "成都"
            }, {
                id: 1012,
                city: "绵阳"
            }, {
                id: 1013,
                city: "宜宾"
            }, {
                id: 1014,
                city: "乐山"
```

```
            }]
        }, {
            id: 102,
            province: "贵州省",
            citys: [{
                id: 1021,
                city: "贵阳"
            }, {
                id: 1022,
                city: "六盘水"
            }, {
                id: 1023,
                city: "黔南"
            }]
        }, {
            id: 103,
            province: "云南省",
            citys: [{
                id: 1031,
                city: "昆明"
            }, {
                id: 1032,
                city: "大理"
            }, {
                id: 1033,
                city: "丽江"
            }]
        }];

        let provinceSelect = document.querySelector('#province');
        let citySelect = document.querySelector('#city');

        let proItem = '';
        let cityMap = new Map();
        data.forEach(item => {
            proItem += `<option value="${item.id}">${item.province}</option>`;
            cityMap.set(item.id, item.citys);
        })
        provinceSelect.innerHTML += proItem;

```

```
        provinceSelect.onchange = function () {
            let citys = cityMap.get(this.value * 1);
            let cityItem = '';

            citys.forEach(item => {
                cityItem += `<option value="${item.id}">${item.city}</option>`;
            })
            citySelect.innerHTML = cityItem;
        }

    </script>
```

9、放大镜

```
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        .demo {
            width: 605px;
            height: 300px;
            display: flex;
            justify-content: space-between;
        }

        .small {
            width: 300px;
            position: relative;
            height: 300px;
            background: url(../img/cat.jpg) 0 0 /cover;
        }

        .mask {
            width: 100px;
            height: 100px;
            position: absolute;
```

```
            background-color: rgba(255, 192, 203, .5);
            text-align: center;
            cursor: move;
            line-height: 100px;
            display: none;
        }

        .big {
            width: 300px;
            height: 300px;
            overflow: hidden;
            display: none;
            position: relative;
        }

        .big img {
            position: absolute;
        }
    </style>

</head>

<body>
    <div class="demo">
        <div class="small">
            <div class="mask">o_0</div>
        </div>
        <div class="big">
            <img src="../img/cat.jpg" alt="" class="bigImg">
        </div>
    </div>

    <script>
        let small = document.querySelector('.small');
        let mask = document.querySelector('.mask');
        let big = document.querySelector('.big');
        let bigImg = document.querySelector('img.bigImg');

        small.addEventListener('mouseenter', function () {
            mask.style.display = 'block';
```

```
            big.style.display = 'block';
        })
        small.addEventListener('mouseleave', function () {
            mask.style.display = 'none';
            big.style.display = 'none';
        })

        small.onmousemove = function (e) {
            let x = e.clientX - mask.clientWidth / 2;
            let y = e.clientY - mask.clientHeight / 2;
            if (x >= small.clientWidth - mask.clientWidth) {
                x = small.clientWidth - mask.clientWidth;
            }
            if (x <= 0) {
                x = 0;
            }
            if (y >= small.clientHeight - mask.clientHeight) {
                y = small.clientHeight - mask.clientHeight;
            }
            if (y <= 0) {
                y = 0;
            }
            console.log(x, y);

            mask.style.left = x + 'px';
            mask.style.top = y + 'px';

            bigImg.style.left = -2 * x + 'px'
            bigImg.style.top = -2 * y + 'px'
        }
    </script>
```